Cycle-CNN for Colorization towards Real Monochrome-Color Camera Systems

Xuan Dong,1 Weixin Li,2* Xiaojie Wang,1 Yunhong Wang2
1School of Computer Science, Beijing University of Posts and Telecommunications, Beijing, 100876
2Beijing Advanced Innovation Center for Big Data and Brain Computing, Beihang University, Beijing, 100191
{dongxuan8811, xjwang}@bupt.edu.cn, {weixinli, yhwang}@buaa.edu.cn

Abstract
Colorization in monochrome-color camera systems aims to colorize the gray image \( I_G \) from the monochrome camera using the color image \( I_C \) from the color camera as reference. Since monochrome cameras have better imaging quality than color cameras, the colorization can help obtain higher quality color images. Related learning based methods usually simulate the monochrome-color camera systems to generate the synthesized data for training, due to the lack of ground-truth color information of the gray image in the real data. However, the methods that are trained relying on the synthesized data may get poor results when colorizing real data, because the synthesized data may deviate from the real data. We present a new CNN model, named cycle CNN, which can directly use the real data from monochrome-color camera systems for training. In detail, we use the colorization CNN model to do the colorization twice. First, we colorize \( I_G \) using \( I_C \) as reference to obtain the first-time colorization result \( I_{C\prime} \). Second, we colorize the de-colored map of \( R_C \), i.e. \( R_G \), using the first-time colorization result \( I_{C\prime} \) as reference to obtain the second-time colorization result \( R_{C\prime} \). In this way, for the second-time colorization result \( R_{C\prime} \), we use the original color map \( R_C \) as ground-truth and introduce the cycle consistency loss to push \( R_C \approx R_{C\prime} \). Also, for the first-time colorization result \( I_{C\prime} \), we propose a structure similarity loss to encourage the luminance maps between \( I_G \) and \( I_{C\prime} \) to have similar structures. In addition, we introduce a spatial smoothness loss within the colorization CNN model to encourage spatial smoothness of the colorization result. Combining all these losses, we could train the colorization CNN model using the real data in the absence of the ground-truth color information of \( I_C \). Experimental results show that we can outperform related methods largely for colorizing real data.

Introduction
With the increasing use of monochrome-color camera systems in high-end smart phones, e.g. Huawei P30, Mate30, etc., the colorization problem within these systems is attracting more and more attentions from the academic and industrial communities.

As shown in Fig. 1, colorization in monochrome-color camera systems aims to colorize the gray image \( I_G \) from the monochrome camera using the color image \( I_C \) from the color camera as reference. Between the monochrome and color cameras, there exist different hardwares, e.g. the color filter array, and different software modules, e.g. white balance, demosaic, etc. As a result, on the one hand, the monochrome camera has better light efficiency (Jeon et al. 2016; Dong et al. 2019) than the color camera, and thus the gray image has higher quality, i.e. signal-noise ratio, than the color image. This motivates researchers to do the colorization so as to get higher quality color images using the monochrome-color camera systems. On the other hand, the pair of gray and color images have different luminance, blur, noises, etc. These cause the difficulty for the colorization.

Among existing methods for colorization within the monochrome-color camera system, some are traditional hand-crafted methods, e.g. (Jeon et al. 2016). With the successful use of deep learning in various computer vision problems, some deep learning based methods, e.g. (Dong et al. 2019) are proposed recently, which have shown to be able to obtain higher accuracy than the traditional ones. However, in the deep learning methods, e.g. (Dong et al. 2019), the models usually need ground-truth color information of the input gray images as annotations for training. Due to the lack of ground-truth color information in the real data, as shown in Fig. 2, current methods, e.g. (Jeon et al. 2016; Dong et al. 2019), usually synthesize data to simulate the
Figure 2: How the synthesized data and the real data are obtained. The real data are the pair of gray and color images shot from the monochrome-color camera system. The synthesized data are the pair of gray and color images that are synthesized using a pair of color images from the dual-color camera system.

Figure 3: The insight of cycle colorization consistancy. When doing the colorization twice, i.e. firstly \((I_G, R_C) \rightarrow I_C\) and secondly \((R_G, I_C) \rightarrow R_C\), the second-time colorization result \(R_C'\) should arrive back at \(R_C\).

real data from the monochrome-color camera system. However, the degradation models for synthesizing the data may deviate from the ones in real imaging systems within the monochrome and color cameras. Thus, the synthesized data could hardly simulate the real data perfectly. As a result, the deep learning methods, which are trained relying on the synthesized data, may have very poor results when colorizing the real data from the monochrome-color system. In detail, as shown in Fig. 4, we use the colorization CNN to do the colorization twice, i.e. firstly colorizing \(I_C\) using \(R_C\) as reference and secondly colorizing \(R_G\) using \(I_C\) as reference. Our objective have three terms, i.e. structure similarity loss, cycle consistency loss, and spatial smoothness loss. For the first-time colorization result \(I_C\), we introduce the structure similarity loss to encourage the structure similarity of the luminance maps between \(I_G\) and \(I_C\). For the second colorization result \(R_C\), we introduce the cycle consistency loss to encourage the similarity of the color maps between \(R_C\) and \(R_C'\). In addition, we introduce the spatial smoothness loss to encourage the spatial smoothness of the colorization result. We train the colorization CNN model by combining all these losses. We also use a refinement CNN to refine \(I_C\) with \(I_G\) as guidance to get the final result \(I_C'\).

Experimental results show that we can outperform related methods largely for the real data from the monochrome-color camera system.

Our contributions include 1) the cycle CNN structure that enables to train the model using real data from the monochrome-color camera systems, 2) the cycle consistency loss for the second-time colorization result, 3) the structure similarity loss for the first-time colorization result, and 4) the spatial smoothness loss for spatial smoothness of the colorization result.

Related Works

The existing colorization tasks can be divided into four kinds, i.e. automatic colorization, scribble-based colorization, reference-based colorization, and monochrome-color dual-lens colorization.

In automatic colorization, the input is only a single gray image and the algorithms need to automatically colorize it without any reference. Recent deep learning based methods,
e.g. (Zhang, Isola, and Efros 2016) and (Iizuka, Simo-Serra, and Ishikawa 2016), make great progress to solve this problem. However, these methods are not proper for our problem because they fail to make use of the color image from the color camera, which contains much useful color information.

In the scribble-based colorization task, the input includes a single gray image and several color scribbles which are drawn by humans. And the methods, e.g. (Zhang et al. 2017) and (Levin, Lischinski, and Weiss 2004), use the color scribbles as guidance to propagate the colors to the whole image. These methods are not suitable for our problem because there exist no scribbles in the monochrome-color camera system.

In the reference-based colorization task, the input includes a single gray image and a reference color image. Different from our problem, the reference image is shot in different locations and/or at different time and the contents within the pair of images just share similar semantics. Because the inputs are different from ours, the methods, e.g. (Welsh, Ashikhmin, and Mueller 2002; Ironi, Cohen-Or, and Lischinski 2005; Gupta et al. 2012; Furusawa et al. 2017; He et al. 2017; 2018; 2019), usually firstly search semantically similar pixels between the images and then propagate the colors of the matching pixels to the whole image. Welsh et al. (Welsh, Ashikhmin, and Mueller 2002) assume that pixels with the same grayscale intensity will have the same color, and use the luminance value as the feature to search for matching pixels. Ironi et al. (Ironi, Cohen-Or, and Lischinski 2005) use discrete cosine transform coefficients as the feature to search sparse matching pixels, copy the color of matching pixels for pixels in high confidence regions and then colorize pixels in low confidence regions by color propagation (Levin, Lischinski, and Weiss 2004). Gupta et al. (Gupta et al. 2012) extract features of superpixels by averaging feature values of all pixels among each superpixel, search for matching pixels by feature matching and use space voting for spatial consistency. Furusawa et al. (Furusawa et al. 2017) propose a reference-based colorization algorithm for colorizing manga images. The assumption for manga images are not always correct for general images. Thus, their results are not always good enough for solving our problem. He et al. (He et al. 2018; 2019) and Zhang et al. (Zhang et al. 2019) propose deep learning based algorithms for image and video colorization. But, they assume the pair of images are visually very different but semantically similar. Due to different assumptions from our problem, they do not consider locality and spatial smoothness and the proposed loss minimizes the semantic differences. Due to different loses, their results are not always faithful to the correct colors.

The monochrome-color dual-lens colorization task can be seen as a special case of reference-based colorization. Jeon et al. (Jeon et al. 2016) propose a stereo matching method to search for best-matching pixels, and correct colors in occlusion regions by applying spatial consistency of neighboring pixels over the whole image. Dong et al. (Dong et al. 2019) proposed a deep CNN for solving this problem and could achieve higher accuracy. However, they rely on synthesized data to train the model. As discussed in the Introduction Section, the real data are quite different from the synthesized data, and their performances on the real data have big decrease.

Like sparseness, smoothness, etc., cycle consistency is also a marvelous and general property and can be utilized for solving different vision problems, e.g. image translation (Zhu et al. 2017), visual tracking (Wang, Jabri, and Efros 2019), super resolution (Wang et al. 2019), etc. In this paper, we make use of it for solving the colorization problem towards real monochrome-color camera systems.

Besides colorization, there exist some other enhancement problems in the multiple-camera system, like video retargeting (Li et al. 2018), deblur (Zhou et al. 2019), style transfer (Chen et al. 2018), etc. But, these methods cannot be directly used for our problem.

The difficulty of collecting ground-truth maps for training is also met in some other vision problems, e.g. depth estimation (Godard, Aodha, and Brostow 2017), image translation (Zhu et al. 2017), etc. And supervised learning methods are not always the most suitable solution. We share similar insights with these methods and propose a self-supervised colorization method in this paper.

**Method**

Our goal is to learn the colorization CNN model \( M \) which maps any given pair of gray and reference color images \( A_G \) and \( B_C \) to the colorization result \( A_C \), i.e. \( M : \{ A_G, B_C \} \rightarrow A_C \). The training data are the real data from monochrome-color camera systems, as shown in Fig. 2.

As shown in Fig. 4, our cycle CNN framework does the colorization using the colorization CNN \( M \) twice. First, we colorize the input gray image \( I_G \) using the color image \( R_C \) as reference. Second, we colorize the de-colored image of \( R_C \), named \( R_G \), using the first-time colorization result \( I_C \) as reference. During the second-time colorization, we use the horizontally flipped maps of \( R_G \) and \( I_C \) to input the colorization CNN \( M \) and do the flip for the result again to get the second-time colorization result \( R_C' \). It is because the model \( M \) always search colors of pixels in the range of \((j, i)\) to \((j, i + d - 1)\) in the reference image for each pixel \((j, i)\) in the input gray image, and the corresponding pixels in \( I_C \) locate in the opposite search range, i.e. from \((j, i)\) to \((j, i - d + 1)\). By doing the flip operations, we can enable the model \( M \) to perform the second-time colorization without changing any model structure. Within the colorization CNN model, for any given pair of gray and reference color
images $A_G$ and $B_C$, we follow (Dong et al. 2019) to learn the 3-D weight volume $W$ between pixels of $A_G$ and $B_C$. And the colorization result $A_C$ is obtained by the weighted average operation between $W$ and $B_C$. In the colorization CNN model, our objective have three terms, i.e. structure similarity loss, cycle consistency loss, and spatial smoothness loss. The structure similarity loss is to encourage the structure similarity of the 3-D weight volume in the colorization CNN. The cycle consistency loss is to encourage the similarity of the color maps between the reference color image $R_C$ and the second-time colorization result $R_C$. The spatial smoothness loss is to encourage the spatial smoothness of the colorization result.

We also use the refinement CNN to refine the first-time colorization result $I_C$ with $I_G$ as guidance to get the final result $I_C$.

Structure similarity loss

We propose the structure similarity loss to measure the structure similarity between the luminance maps of $I_C$ and $I_G$, i.e. $I^{s}_{G}$ and $I^{s}_{C}$. Our insight is that if the colorization is perfect, not only the color information, i.e. $Cb$ and $Cr$, should be correct, but also the luminance maps should have similar structures. $I^{s}_{G}$ and $R^{s}_{C}$ have different intensities, due to different luminance, blur, noise, etc., during the imaging process of the monochrome and color cameras. So, traditional loss metrics, e.g. $L1$ loss, $L2$ loss, etc., are not effective for our case.

We propose a CNN model to learn a deep metric to obtain the structure similarity loss between $I^{s}_{G}$ and $I^{s}_{C}$. As shown in Fig. 5, we use two ResNet to extract the features of $I^{s}_{G}$ and $I^{s}_{C}$, respectively. In addition, motivated by the success of SSIM (Wang et al. 2004), we also extract the luminance map $l$, the contrast map $c$ and the structure map $s$ between $I^{s}_{G}$ and $I^{s}_{C}$ as the features (please refer to more details in (Wang et al. 2004)). Then, we feed these five features to a ResNet to estimate the final structure similarity map $SSM$, and the structure similarity loss is obtained by

$$L_{ss} = 1 - \mu(SSM(I_{G}, I_{C})), \tag{1}$$

where $\mu$ is the average operation.

To train the CNN model, we synthesize a dataset. First, we use the monochrome-color dual-camera system to shoot pairs of images in the distant view. Second, we use SIFT feature and projective transform to register the pairs of image. Among the registered results, we manually select and crop out the sub-regions where all pixels are perfectly registered and there exist no occlusions at all. The sub images are named $J_{G}$ and $J_{C}$. Then, to simulate the structure distortions between $I^{s}_{G}$ and $I^{s}_{C}$, we propose to generate several random warping maps and use these maps to warp $J_{G}$ and $J_{C}$. The warped images are named $\hat{J}_{G}$, $\hat{J}_{C}$.

The pair of $\hat{J}_{C}$ and $J_{G}$ are used as the input, and the SSIM quality map (Wang et al. 2004) between $\hat{J}_{G}$ and $J_{C}$ is used as the ground-truth structure similarity map. We train the model using $L1$ loss. The random seeds include randomly values with sizes of $3 \times 3$, and we interpolate the seeds to the size of $J_{G}$ using Bicubic to obtain the random warping maps. The mean and variance of the random values of the seeds are 0 and $\{1, 5, 10, 20, 40\}$.

In our colorization CNN model, we use the pre-trained structure similarity network to generate the structure similarity loss.

Cycle consistency loss

To encourage the similarity of the color maps between the reference color image $R_{C}$ and the second-time colorization result $R_{C}$, we propose the cycle consistency loss to measure the differences between $R_{C}$ and $R_{C}$. We use SSIM as the metric and we measure both $Cb$ and $Cr$ color channels, i.e.

$$L_{cc} = 1 - \frac{1}{2}(SSIM(R^{Cb}_{C}, R^{Cb}_{C}) + SSIM(R^{Cr}_{C}, R^{Cr}_{C})), \tag{2}$$

Spatial smoothness loss

We introduce the spatial smoothness loss to encourage spatial smoothness of the $W$ volume in the colorization CNN $M$ so as to obtain spatial smooth colorization result. We assume that neighboring pixels should have similar weights, so the loss is defined as

$$L_{smooth} = \frac{1}{N} \sum_{(j,i,k)} \sum_{(j',i',k') \in \Omega(j,i,k)} |W_{j,i,k} - W_{j',i',k'}|, \tag{3}$$

where $\Omega$ is the 6-neighboring pixels in the three dimensions.

Full objective

Combining all above losses, the overall objective we aim to optimize is:

$$L = \lambda_1 L_{ss} + \lambda_2 L_{cc} + \lambda_3 L_{smooth}, \tag{4}$$

where $\lambda_1$, $\lambda_2$, and $\lambda_3$ control the relative importance of the corresponding terms respectively. The values are set as $\lambda_1 = 1$, $\lambda_2 = 1$, and $\lambda_3 = 0.1$ in this paper. With the guidance...
of these losses, we successfully learn the colorization CNN without the ground-truth color information for training.

**Color refinement**

The first-time colorization result $I_C$ may have errors in occlusion regions. To correct these errors, we use the input gray image $I_G$ as guidance to refine the colorization result. We follow (Dong et al. 2019) to build the network. As shown in Fig. 7, the input gray image $I_G$ is fed into a ResNet to get its feature. The extracted feature and $I_C$ are concatenated and then fed into another ResNet to get the residue color map $\Phi(I_C, I_G)$. By adding $I_C$ and $\Phi(I_C, I_G)$, the final colorization result $I'_C$ is obtained. To train this model, we use the second-time colorization results $R'_C$ as inputs and the original input color image $R_C$ as ground-truth, and we use L2 loss for training.

**Network architecture**

We follow (Dong et al. 2019) to build the colorization CNN model and refinement CNN model. The pipeline is shown in Figs. 6 and 7. The ResNet has 18 convolution layers in total. The first residue block is with $5 \times 5$ kernel. The following 16 layers are 8 repeated residue blocks and each residue block consists of 2 convolution layers with $3 \times 3$ kernel and a residue connection. $BatchNorm$ layers and $ReLu$ layers are added after each convolution layer. The filter number $n$ of the 18 layers of ResNet is a hyper-parameter, which is set as 16 in this paper. 2) In the refinement module, the ResNets have similar network structure. The difference is that in the last layer the filter number is 1 and no $BatchNorm$ layer or $ReLu$ layer is added.

**Experimental Results**

**Dataset**

We use one monochrome camera and one color camera to shoot 1000 pairs of gray and color images to build the dataset, named Real Dataset. The monochrome and color cameras are rectified using the method of (Bradski and Kaehler 2008). The cameras are the monochrome and color versions of the same camera, i.e. the MVCAM-SU1000C camera.

**Implementation details**

The proposed deep convolutional network is implemented with TensorFlow. All models are optimized with RMSProp and a constant learning rate of 0.001. We train with a batch size of 1 using a $256 \times 512$ randomly located crop from the input images. The images of the dataset is randomly divided into the training set, which contains 700 pairs of images, and the testing set, which contains 300 pairs of images. All the models are run on a server with an Intel I7 CPU and 4 NVIDIA Titan-X GPUs. The training time is about 15 hours and the testing time is about 0.4 seconds for $780 \times 1024$ test images.

**Comparison algorithms**

We compare with state-of-the-art reference-based colorization algorithms, i.e. Welsh et al. (Welsh, Ashikhmin, and Mueller 2002), Iironi et al. (Iironi, Cohen-Or, and Lischinski 2005), Gupta et al. (Gupta et al. 2012), Furusawa et al. (Furusawa et al. 2017), He et al. 2018 (He et al. 2018), and He et al. 2019 (He et al. 2019), automatic colorization algorithms, i.e. Zhang et al. (Zhang, Isola, and Efros 2016) and Iizuka et al. (Iizuka, Simo-Serra, and Ishikawa 2016), and monochrome-color dual-lens colorization algorithms, i.e. Jeon et al. (Jeon et al. 2016) and Dong et al. (Dong et al. 2019).

**Comparison with other colorization methods on Real Dataset**

The qualitative results are shown in Figs. 8 and 9. As shown, our method has better results than the comparison methods. The automatic colorization methods (Iizuka, Simo-Serra, and Ishikawa 2016; Zhang, Isola, and Efros 2016) have wrong colors in most regions. It is because the input in these methods is only one single gray image, and the reference color image, which could provide much useful color information during the colorization, is not utilized at all. Welsh et al.’s method does not have good performance, because their assumption, i.e. pixels with the same grayscale intensity will have the same color value, is not true for many images. Gupta et al.’s method does not perform well, especially for objects with complicated textures. It is because the features of each superpixel are obtained by averaging the feature values of all pixels in the superpixel, which will
Figure 9: Examples to compare the colorization results of the monochrome-color dual-lens colorization methods of Jeon et al., and Dong et al., and the reference-based colorization methods of Ironi et al., He et al. 2018, He et al. 2019, with ours.

Figure 10: Example results in the ablation study.

Figure 11: Examples of our first-time colorization results and second-time colorization results.

Table 1: Average SSIM values of the second-time colorization results of different methods on the Real Dataset.

A user study is also performed. There are 30 annotators in total. The annotation choices include five score level, i.e. ‘Perfect’, ‘Few Errors’, ‘Partly Wrong’, ‘Mostly Wrong’, and ‘Totally Wrong’. The annotators are asked to annotate every colorization result of the 10 comparing methods and ours. The whole set of images in the user study are 100 pairs that are randomly selected from our Real Dataset. And each annotator annotates 1100 colorization results in total. To avoid outlier annotation, we will let each annotator randomly re-annotate some results and see the annotations as outlier if the annotation differences are beyond one score level. The results are shown in Fig. 12. This shows we can get ‘Perfect’ and ‘Few Errors’ scores in most cases and our method gets much higher perceptual scores than the others. **Objective evaluation** is also performed by evaluating...
Table 2: Average SSIM values of the first-time colorization results and the second-time colorization results of different methods on the four synthesized datasets. CT, MB, ST, and SF are short for the datasets of Cityscapes, Middlebury, Sintel, and SceneFlow, respectively.

<table>
<thead>
<tr>
<th></th>
<th>First-time colorization</th>
<th>Second-time colorization</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CT</td>
<td>MB</td>
</tr>
<tr>
<td>Welsh</td>
<td>0.897</td>
<td>0.906</td>
</tr>
<tr>
<td>Ironi</td>
<td>0.897</td>
<td>0.940</td>
</tr>
<tr>
<td>Gupta</td>
<td>0.908</td>
<td>0.986</td>
</tr>
<tr>
<td>Jeon</td>
<td>0.933</td>
<td>0.958</td>
</tr>
<tr>
<td>Furusawa</td>
<td>0.841</td>
<td>0.860</td>
</tr>
<tr>
<td>He18</td>
<td>0.951</td>
<td>0.949</td>
</tr>
<tr>
<td>Zhang</td>
<td>0.986</td>
<td>0.986</td>
</tr>
<tr>
<td>Iizuka</td>
<td>0.737</td>
<td>0.677</td>
</tr>
<tr>
<td>Dong</td>
<td>0.984</td>
<td>0.981</td>
</tr>
<tr>
<td>He19</td>
<td>0.992</td>
<td>0.993</td>
</tr>
<tr>
<td>Ours</td>
<td>0.982</td>
<td>0.983</td>
</tr>
</tbody>
</table>

Comparison with other colorization methods on Synthesized Dataset

Due to the importance of objective evaluation, we perform our method on the traditional synthesized dataset of (Dong et al. 2019; Jeon et al. 2016). The datasets include Cityscapes (Cordts et al. 2016), Middlebury (Scharstein and Pal 2007), Sintel (Butler et al. 2012), and SceneFlow (Mayer et al. 2016). We use all the comparing methods and ours to do the colorization twice. The results are in Table 1, which show that we outperform largely than the other methods.

Conclusions

We have presented a novel CNN model for colorization in real monochrome-color dual-lens system. It can be trained directly using the real data from monochrome-color camera systems. The proposed method uses the CNN model to do the colorization twice. In addition, we introduce the cycle consistency loss, the structure similarity loss, and the spatial smoothness loss. Our method achieves superior performance than the state-of-the-art methods for colorizing real data.

Acknowledgments

This work is funded by the National Nature Science Foundation of China (No. 61802026 and 61806016) and the Fundamental Research Funds for the Central University (No. 2019RC39). We thank the anonymous reviewers for helping us to improve this paper.

References


Godard, C.; Aodha, O.; and Brostow, G. 2017. Unsupervised monocular depth estimation with left-right consistency. *CVPR*.


